Towards Collaborative Applications using Web Services

Ricardo Coppo Claudio Delrieux

Departamento de Ing. Eléctrica y Computadoras
Universidad Nacional del Sur, Bahía Blanca, ARGENTINA
e-mail: rcoppo@bblanca.com.ar, claudio@acm.org

1 Introduction

Web services are becoming popular as a system integration solution for distributed applications. They allow for publishing, finding and binding of distributed software components over a Wide Area Network like the Internet and are platform and operating system independent.

In a traditional scenario, ontologies are defined and published by a consortium or enterprise group for each application discourse. These specifications can later be implemented by many web service providers.

Consider an institutional service that could be defined by the countries government or board of education. Each participating college or university could later implement the service on a local basis. Users would discover the available services in a directory through some kind of search facility and develop front-end applications. Receiving the data in a standardized format allows the information of each participating institution to be integrated into existing applications without disrupting the normal look and feel that each user experiences on his own terminal.

Collaborative applications require a different approach. Each prospective user is expected to download, process and return results to a centralized agent, collaborating with her participation to the accomplishment of a much larger goal and benefiting from the results that other members supply. The collaborative effort becomes a massively parallel human and computational processing resource devoted to the resolution of complex problems in small segments.

2 Collaborative actors

Three profiles can be defined for the actors participating in the collaborative effort: the collaborative participant or member, the collaborative server and the end-user client.

2.1 The collaborative participant

This user type will typically receive prepackaged software to interact with the collaborative server, download data segments to be processed, and upload results obtained on her machine.

Simple collaborative applications will only request the use of raw computing power from this kind of user, downloading to the participant's machine data segments and harvesting results as fast as the local computer can produce. The collaborative prepackaged software would probably run in a back-ground or idle cycle stealing processing mode so as not to disrupt local processing.

In a more complex scenario more human interaction and decision making would be required from the collaborative member. Applications using image processing, fotointerpretation and classification algorithms are typical examples. In these cases, human intelligence and expertise might be needed to solve the problem at hand. This will make the participants participation a long running process for the server while these decisions are being taken.

The prepackaged software can or may require third party programs to be installed on the local machine to do their work. The collaborative can now take advantage of high cost licensed software installed on the client's machine without infringement to software copyright law.

On occasions experienced software developers may want to write their own interfaces for the collaborative server. The use of web services as a common communication mechanism allows them to access the server in a programmatic way in a platform and operating system independent manner. Most development packages now include special tools to connect and use web services.

2.2 The collaborative server

The collaborative server is responsible for the control of long duration processing threads. Each user, as she logs in creates a new business process workflow and begins interacting with the raw data in a controlled manner. The workflow state is serialized to a special database. As the process advances from state to state the workflow description determines possible alternatives, timeouts and error processing.

The server will maintain the original data bank and serve requests for partial data segments of the raw data. Once the results are received from the collaborative participants they are stored in a final data store.

An operational console should allow the collaborative controller to visualize the amount of open flows, their present processing state and provide options to kill or release unfinished threads.

Other objectives for the server include information security procedures such as user validation, data encryption, and database administration.

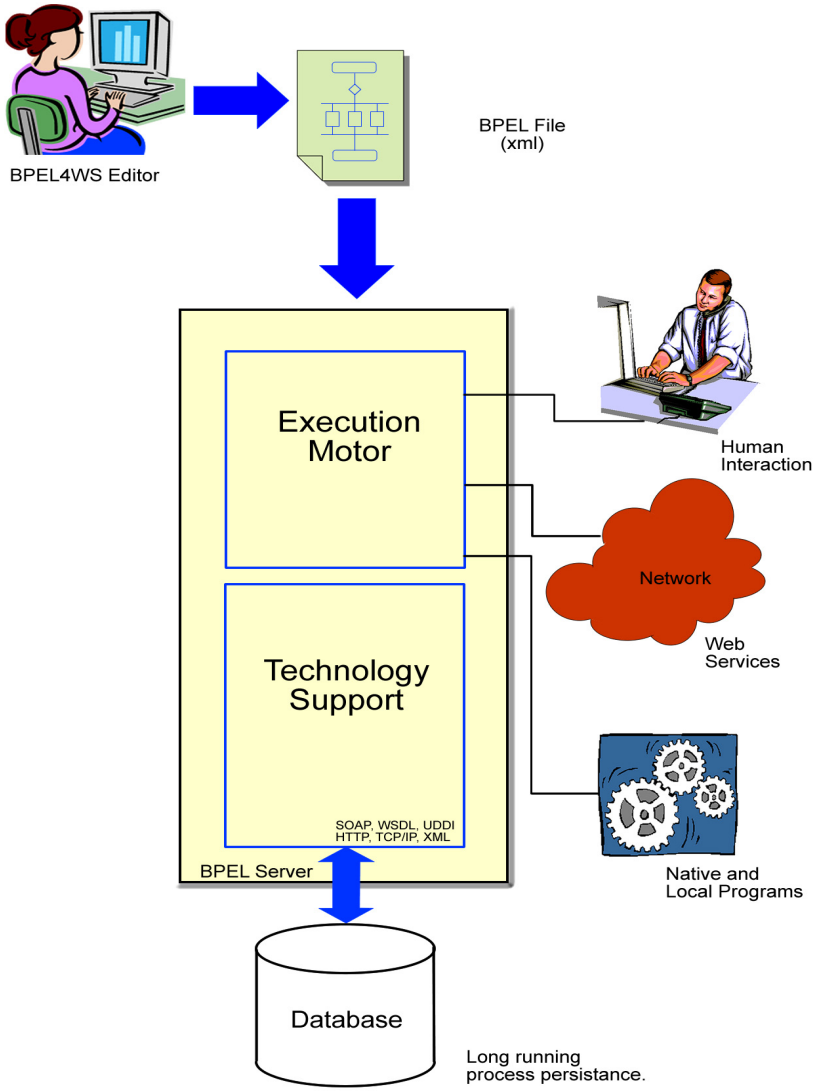

Figure 1: BPEL Architecture

2.3 End-user client

Not all potential users will want to participate in the processing of the raw data. Most will only want the finished results to be available for their own personal use.

These users can be considered as end-user clients and may wish to receive the information through different channels. Typical clients will want a web page interface but more advanced users will wish to develop their own applications. Delivering the processed information through web services will greatly enhance its usability.

3 Workflow Languages

Long running workflows that require state serialization have been common in business applications that move information over enterprise boundaries. (Consider for example a invoice procedure that requires credit checks for the buyer, supplier stock availability, and shipping requests).

These processes can be modeled with different languages and have been recently updated to include web services as a possible means of interaction.

Both IBM and Microsoft have defined workflow languages based on XML. Their individual efforts have been united in BPEL4WS - Business Process Execution Language for Web Services. Many DBMS vendors have incorporated BPEL4WS as extensions to their core products.

Using BPEL4WS the collaborative developer can specify the workflow, with its errors and exceptions in a standard language. He can specify the workflow using a visual editor and submit the resulting XML file directly to the collaborative server.

On the server, a BBEL engine keeps track of the open flows and their present state. If a user tries to break out of the flow either by an intent to access unauthorized services or by not following the correct sequence the engine will abort the flow.

The BPEL engine delivers console output to the collaborative controller and interfaces with other web services available over the network or with local programs when necessary.

The BPEL engine is an abstract concept and may itself be implemented over a host of real machines if it must sustain high workloads or a great number of users.

References

[1] Arsanjani et al., *"Web Services: Promises and Comprimises"*,Queue, pp 49-58, March 2003.

[2] Cauldwell P., Chawla R. et al., *"Professional XML Web Services"*, Wrox Press Ltd., London, 2001.

[3] Cerami E., *"Web Services Essentials"*, O'Reilly, New York, 2002.

[4] Erl T., *"Service Oriented Architecture: A Field Guid to Integrating XML and Web Services"*, Prentice Hall PTR, 2004.

[5] Fremantle S., Weerawarana, Khalaf R., *"Enterprise Services"*, CACM, 45:10, pag. 77-82, October 2002.

[6] Gibbins N., Harris S., Shadbolt N., *"Agent-based Semantic Web Services"*, Proceedings of the 12th WWW, Budapest, Hungría, pag. 710-717, May 2003.

[7] Gottschalk K., Graham S., Kreger H., et al., *"Introduction to Web Services Architecture"*, IBM Systems Journal, 41(2), pag. 170-177, 2002.

[8] IBM, Microsoft, et al., *"Business Process Execution Language for Web Services"*, version 1.1, 2003.

[9] Kleijnen S., Raju S., *"An Open Web Service Architecture"*, Queue, pag 39-46, Marzo 2003.

[10] Leyman, F., et al. *"Web Services Flow Language"* International Business Corporation (IBM), Software Communications Department, Route 100, Building 1, Somers, NY 10589, USA, May 2001.

[11] Newcomer E., *"Understanding Web Services: XML, WSDL, SOAP, and UDDI"*, Addison-Wesley Professional, 2002.

[12] Siegel J., *"Achieving Interoperability for Integration of Heterogeneous COTS Geographic Information Systems"*, Proceedings of ACM GIS 2002, McLean, Virginia, EEUU, pag. 162-167, 2002.

[13] Thatte, S., *"XLANG - Web Services for Business Process Design"*, Microsoft Corporation, 2001.

[14] Visser, Ubbo, *"Intelligent Information Integration for the Semantic Web"*, Springer-Verlag Berlin Heidelberg, 2004.

[15] W3C *"Web Services Architecture"*, W3C Working Group Note 11 February 2004, Editors: David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris David Orchard, World Wide Web Consortium, **http://www.w3.org/TR/ws-arch/**, 2004.

[16] WfMC, *"Workflow Management Coalition Workflow Standard Process Definition Interface - XML Process Definition Language"*, Workflow Management Coalition, 2436 N.Federal Highway #374, Lighthouse Point, FL 33064, USA, Document # WFMC-TC-1025, Version 2.0, 2005.

[17] Yin J., et al., *"Engineering Server-Driven Consistency for Large Scale Dynamic Web Services"*, Proceedings of the 10th international conference on WWW, Hong Kong, pag. 45-57, Mayo 2001.

[18] Zeng L., Benatallah B., Dumas M., *"Quality Driven Web Services Composition"* Proceedings of the 12th International conference on WWW, Budapest, Hungría, pag. 411-421, Mayo 2003.